

Automatic Text Segmentation with String Vector based KNN Variants

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the string vector based KNN variants as the approaches to the text segmentation. The initial KNN algorithm was previously modified into the string vector-based version as an approach to the task. In case of the numerical vector-based versions, we mention the three KNN variants: one where the selected nearest neighbors are discriminated by their similarities, one where the attributes are discriminated by their correlations with the categories, and one where the training examples are discriminated by their credits. The three KNN variants are modified into ones each of which receives a string vector as its input data, as well as the initial KNN algorithm. This research is intended to improve further the text segmentation performance by modifying so.

I. INTRODUCTION

The text segmentation refers to the process of partitioning a single text into subtexts based on the content transition. In the task, which is covered in this research, a single text is given as the input, it is partitioned into paragraphs, a boundary is set between adjacent paragraphs where the topics is changed abruptly. The process of text segmentation is to partition a single text into paragraphs, classify each adjacent paragraph pair into one of boundary and continuance, and set a boundary in the position between paragraphs in each pair which is classified into boundary. The subtexts which are results from segmenting a text are treated as independent texts.

This research is motivated by the successful case of using the string vector based KNN algorithm for the text segmentation in the previous work. The text segmentation was viewed into a binary classification of adjacent paragraph pairs, they were encoded into string vectors, and the semantic operation between string vectors was defined. The KNN algorithm was modified into the string vector-based version and applied to the text segmentation. The string vector-based version was empirically validated as the more recommendable approach to the text segmentation, than the traditional version. In this research, the KNN variants will be modified into the string vector-based versions as the approaches to the text segmentation.

The idea of this research is to modify the KNN variants into the string vector-based versions as the approaches to the text segmentation. The similarity metric between string vectors which is called semantic similarity will be used for

modifying the KNN variants as well as the KNN algorithm. In this research, we propose the three string vector-based KNN variants: one where the nearest neighbors are discriminated by their distances for voting their labels, one where the attributes are discriminated by their correlations with the target output values for computing similarities of a novice item with the training examples, and one where the training examples are discriminated by their qualities for doing both. The text segmentation is mapped into a binary classification of two adjacent paragraphs into boundary or continuance, and the string vector based KNN variants are applied to the task. In this research, the modified KNN variants are initially proposed as the approaches to the text segmentation.

Let us mention some benefits from this research. Encoding texts into string vectors prevents the main problems in encoding texts into numerical vectors. The KNN algorithm is replaced by the KNN variants for improving the classification performance. The KNN variants is modified into the string vector-based versions for improving it in addition. The goal of this research is to provide the more recommendable approaches to the text segmentation.

Let us mention the organization of this paper. In Section II, we explore the previous works which are relevant to this study. In Section III, we describe in detail what is proposed in this research. In Section IV, we mention the significances of this research and the remaining tasks. This paper is composed of the four sections.

II. PREVIOUS WORKS

This section is concerned with the previous works which are relevant to this research. It is intended to expand the style of modifying the KNN algorithm to its three variants as the approaches to text segmentation. The directions of exploring previous works are derivation of KNN variants, modification of the standard KNN algorithm into its string vector-based version as an approach to the task, and the previous case of encoding a text into a string vector. The distinguishable points of this research are derivation of three KNN variants from the standard version, modification of them into their string vector-based versions, and application of them to the text segmentation. This article is intended to explore previous works for providing the background for this research.

Let us explore the previous works on KNN variants. In 2001, the KNN variant where nearest neighbors are discriminated by their distances from or similarities as a novice item was applied to the text classification by Han et al. [1]. In 2008, MKNN (Modified K Nearest Neighbor) the KNN variants where training examples are discriminated by their validness, was proposed by Parvin et al. [2]. In 2018, the KNN variant where the attributes are discriminated by their correlations were proposed by Nababan and Sitompul [10]. However, this research uses different specific metrics for discriminating attributes, nearest neighbors, and training examples.

Let us explore the previous works on applying the modified version of KNN algorithm to the text segmentation. In 2017, it was initially proposed that the string vector-based version of KNN algorithm should be applied to the text segmentation as a position paper by Jo [6]. In 2019, the modified KNN algorithm was validated in the task by Jo [12]. The journal article which describes in detail the modified KNN algorithm and its application to the text segmentation is finally prepared for its publication by Jo, as well as text segmentation [13]. This research is based on the three previous works.

Let us explore the previous works on the neural networks, NTC (Neural Text Categorizer), where encoding a text into a string vector was initially proposed. In 2010, it was initially invented as an approach to the text classification by Jo [3]. In 2015, it was applied to the topic identification of Arabic texts by Abainia [5]. In 2018, it was evaluated as the most practical neural networks by Lakshmi and Rao [9]. In 2010, the NTSO (Neural Text Self Organizer) was also invented as an approach to the text clustering by Jo [4].

Let us mention the differences of this research from the previous works which were explored above. The KNN variants in the previous works are numerical vector-based versions, and they will be modified into string vector-based versions in this research. As the expansion of [7], we modify and adopt the KNN variants for implementing the text segmentation system. Machine learning algorithms will be innovated by inventing various ones which process directly string vectors as their input. The modified KNN variants will be described in detail in Section III.

III. PROPOSED WORKS

This section is concerned with what is proposed in this research. In Section III-A, we describe the process of encoding a text into a string vector and the proposed similarity metric. In Section III-B, we describe the standard version of KNN algorithm where the proposed similarity metric is adopted. In Section III-C, we derive the three variants from the standard version. In Section III-D, we present the system architecture of the text segmentation system.

A. Encoding Proccerss

This section is concerned with the string vector as the text representation, and the similarity between string vectors. A string vector is a finite ordered set of strings; numerical values are replaced by strings as the elements in a vector. It is required to define the similarity matrix for performing the operation on string vectors. The similarity between string vectors is the average over one-to-one similarities between their elements. This section is intended to describe the process of encoding a text into a string and the process of computing the similarity between string vectors.

The process of encoding a text into a string vector is illustrated in Figure 1. A text is indexed into a list of words and their information. The features of a string vector are defined as symbolic forms manually in the feature definition. A string which represents a text is filled with the words which corresponds to the features in the feature value assignment. Refer to [11] for studying the encoding process in more detail.

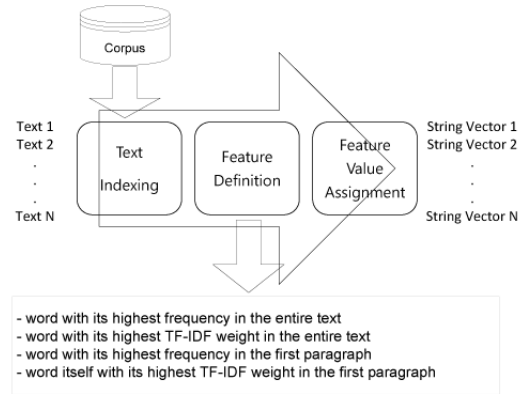


Figure 1. Process of Encoding Texts into String Vectors

The similarity matrix which is the basis for computing the similarity between two string vectors is illustrated in Figure 2. Each row and each column correspond to their own word, and each element in the similarity matrix is the similarity between two words, t_1 and t_2 , as shown in equation (1),

$$s_{ij} = \text{sim}(t_i, t_j) \quad (1)$$

The similarity between two words, t_i and t_j is computed by equation (2),

$$\text{sim}(d_i, d_j) = \frac{2\#(t_i \wedge t_j)}{\#(t_i) + \#(t_j)} \quad (2)$$

where $\#(t_i \wedge t_j)$ is the number of texts which include both t_i and t_j , $\#(t_i)$ is the number of texts which includes t_i , and $\#(t_j)$ is the number of texts which includes t_j . The value of similarity between two words, t_i and t_j , is always given as a normalized value between zero and one, as shown in equation (3),

$$0 \leq \text{sim}(t_i, t_j) \leq 1 \quad (3)$$

The similarity matrix is used for computing the similarity between string vectors which represent texts as the reference table.

$$\begin{array}{c} \text{Word 1} \quad \text{Word 2} \quad \dots \quad \text{Word N} \\ \begin{bmatrix} \text{Word 1} & S_{11} & S_{12} & \dots & S_{1N} \\ \text{Word 2} & S_{21} & S_{22} & \dots & S_{2N} \\ \dots & \dots & \dots & \dots & \dots \\ \text{Word N} & S_{N1} & S_{N2} & \dots & S_{NN} \end{bmatrix} \end{array} \quad s_{ij} = \frac{2 \times \#(\text{word}_i, \text{word}_j)}{\#(\text{word}_i) + (\text{word}_j)}$$

Figure 2. Semantic Similarity Matrix

Let us mention the process of computing the similarity between string vectors as a semantic similarity between texts. The two string vectors which represent texts are notated by $\mathbf{str}_1 = [t_{11} \ t_{12} \ \dots \ t_{1d}]$ and $\mathbf{str}_2 = [t_{21} \ t_{22} \ \dots \ t_{2d}]$. The similarity between two string vectors is computed by equation (4),

$$\text{sim}(\mathbf{str}_1, \mathbf{str}_2) = \frac{1}{d} \sum_{i=1}^d \text{sim}(t_{1i}, t_{2i}). \quad (4)$$

The similarity between elements, $\text{sim}(t_{i1}, t_{2i})$, is taken from the similarity which was mentioned above. The value which is generated by equation (4) is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq \text{sim}(\mathbf{str}_1, \mathbf{str}_2) \leq 1. \quad (5)$$

Let us make some remarks on the encoding process and the computation of the similarity between texts. A text is encoded into a string vector by the word-text association, the feature definition, and the feature value assignment. The similarity matrix is defined as the basis for computing the similarity between two string vectors. The similarity between two string vectors is computed by equation (4). In the future research, we consider representing a text into multiple string vectors.

B. Initial Version of String Vector based KNN Algorithm

This section is concerned with the initial version of string vector based KNN algorithm. It was initially proposed as the approach to the text summarization by Jo in 2018 [8]. The similarity metric between string vectors which was described in the previous section is used for computing the similarity between a novice string vector and a training example. The labels of its nearest neighbors are voted with their identical weights for deciding the label of a novice input. This section is intended to describe the initial version of KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 00. The labeled words which are gathered as samples are encoded into string vectors, and they are notated by a set, $Tr = \{\mathbf{str}_1, \mathbf{str}_2, \dots, \mathbf{str}_N\}$. A novice

word is encoded into a string vector notated by $\mathbf{str}$. Its similarities with the string vectors which represent the sample words are computed by equation (4), as $\text{sim}(\mathbf{str}, \mathbf{str}_1), \text{sim}(\mathbf{str}, \mathbf{str}_2), \dots, \text{sim}(\mathbf{str}, \mathbf{str}_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

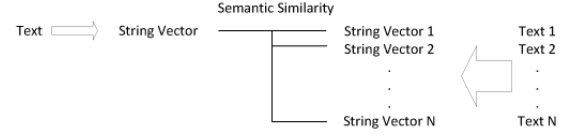


Figure 3. Similarities of Novice Input with Training Examples

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice example are selected as its nearest neighbors, as expressed in equation (6),

$$Nr = \text{Select}_k(Tr, \mathbf{str}), Nr \subseteq Tr \quad (6)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (7),

$$Nr = \{\mathbf{str}_1^{near}, \mathbf{str}_2^{near}, \dots, \mathbf{str}_k^{near}\} \quad (7)$$

The elements in the set, Nr , are used for voting their labels in the next step.

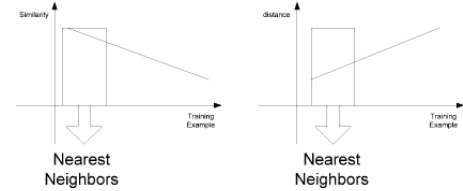


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = \text{label}(\mathbf{str}_i^{near})$. The number of nearest neighbors which belong to the category, c_j , is notated by $\text{Count}(Nr, c_j)$. The label of the novice item, $\mathbf{str}$, is decided by equation (8),

$$\text{label}(\mathbf{str}) = \arg\max_{j=1}^m \text{Count}(Nr, c_j) \quad (8)$$

The process of deciding the label of a novice item by equation (8), is called voting.

Let us make some remarks on the initial version of KNN algorithm from which we derive some variants. It computes the similarities of a novice item with the training examples.



Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

The training examples with their highest similarities with the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. KNN Variants

This section is concerned with the variants which are derived from the KNN algorithm which was covered in Section III-B. The difference from the traditional version is to adopt similarity metric between string vectors for computing the similarity between a novice item and a training example. In this section, we derive the three variants by discriminating nearest neighbors, attributes, or training examples. The three variants of KNN algorithm are adopted for implementing the text segmentation system. This section is intended to describe the three variants.

Let us mention the KNN variant which is derived by discriminating the nearest neighbors. A set of nearest neighbors is notated by $Nr = \{\mathbf{str}_1^{near}, \mathbf{str}_2^{near}, \dots, \mathbf{str}_k^{near}\}$, and its elements are assumed as $\text{sim}(\mathbf{str}, \mathbf{str}_1^{near}) \geq \text{sim}(\mathbf{str}, \mathbf{str}_2^{near}) \geq \dots \geq \text{sim}(\mathbf{str}, \mathbf{str}_k^{near})$. The weights, $w_1, w_2, \dots, w_k$, are assigned to the nearest neighbors, $\mathbf{str}_1^{near}, \mathbf{str}_2^{near}, \dots, \mathbf{str}_k^{near}$, following, $w_1 \geq w_2 \geq \dots \geq w_k$, and the total weights are computed for the category, c_j by equation (9),

$$CS(Nr, c_j) = \sum_{\mathbf{str}_i \in Nr} w_i \quad (9)$$

The label of the novice item, $\mathbf{str}$, is decided by equation (10),

$$\text{label}(\mathbf{str}) = \arg\max_{j=1}^m CS(Nr, c_j) \quad (10)$$

There are two ways of assigning weights to the nearest neighbors: exponentially decreasing way and arithmetic decreasing way as shown in Figure 6.

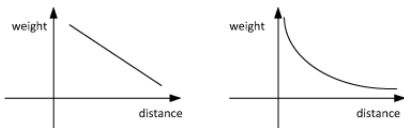


Figure 6. Discrimination over Nearest Neighbors

Let us mention the second KNN variant where the attributes are discriminated for computing the similarity

between a novice item and a training example as shown in Figure 7. If the number of training examples is N , and the dimension of numerical vector representing a training example is d , the attributes are notated as a set, $A = \{A_1, A_2, \dots, A_d\}$, and each attribute is viewed as a set of N values $A_i = \{str_{1i}, str_{2i}, \dots, str_{Ni}\}$. The occurrences of the string value, str_{ij} , in the category, c_k , is notated by $f(str_{ij}|c_k)$. The influence of the attribute, A_i , on the category, c_k , is computed by equation (14),

$$I(A_i, c_k) = \frac{1}{N} \sum_{j=1}^N f(str_{ji}, c_k), \quad (11)$$

and the weight of the attribute, A_i , is computed by equation (15),

$$w_i = \max_{k=1}^m I(A_i, c_k). \quad (12)$$

The similarity between the novice input, $\mathbf{str}$ and the training example, $\mathbf{str}_i$, is computed by equation (16),

$$\text{sim}(\mathbf{str}_i, \mathbf{str}) = \frac{1}{d} \sum_{j=1}^d w_i \text{sim}(str_{ij}, str_j) \quad (13)$$

In this variant, the equation (4) is replaced by equation (16) for computing the similarity between a novice item and a training example.

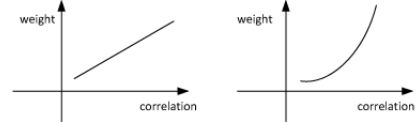


Figure 7. Discrimination over Attributes

Let us mention the second KNN variant where the attributes are discriminated for computing the similarity between a novice item and a training example as shown in Figure 7. If the number of training examples is N , and the dimension of numerical vector representing a training example is d , the attributes are notated as a set, $A = \{A_1, A_2, \dots, A_d\}$, and each attribute is viewed as a set of N values $A_i = \{str_{1i}, str_{2i}, \dots, str_{Ni}\}$. The occurrences of the string value, str_{ij} , in the category, c_k , is notated by $f(str_{ij}|c_k)$. The influence of the attribute, A_i , on the category, c_k , is computed by equation (14),

$$I(A_i, c_k) = \frac{1}{N} \sum_{j=1}^N f(str_{ji}, c_k), \quad (14)$$

and the weight of the attribute, A_i , is computed by equation (15),

$$w_i = \max_{k=1}^m I(A_i, c_k). \quad (15)$$

The similarity between the novice input, $\mathbf{str}$ and the training example, $\mathbf{str}_i$, is computed by equation (16),

$$\text{sim}(\mathbf{str}_i, \mathbf{str}) = \frac{1}{d} \sum_{j=1}^d w_i \text{sim}(str_{ij}, str_j) \quad (16)$$

In this variant, the equation (4) is replaced by equation (16) for computing the similarity between a novice item and a training example.

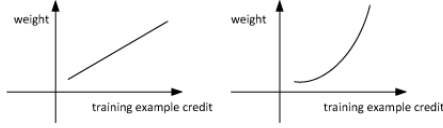


Figure 8. Discrimination over Training Examples

Let us make some remarks on the three variants of KNN algorithm which are proposed as the approaches to the text segmentation. In the first variant, the nearest neighbors are discriminated for voting their labels. In the second variant, the attributes are discriminated for computing the similarity between a novice input and a training example. In the third variant, the training examples are discriminated for voting the labels of the nearest neighbors and/or computing the similarity between string vectors. We may consider the trainable KNN algorithm which optimizes the weights of the nearest neighbors, the attributes, and the training examples for minimizing the training error.

D. System Architecture

This section is concerned with the architecture and the execution flow of the text segmentation system. In Section III-C, we describe the string vector based KNN variants and adopt them for implementing the text segmentation system. The initial KNN algorithm which is mentioned in Section III-B is replaced by its variants in the architecture of the text segmentation system which was previously proposed. The process of executing the system is to encode an adjacent paragraph pair into a string vector and classify it into boundary or continuance. This section is intended to describe the text segmentation system which is implemented in this study.

The process of gathering sample paragraph pairs and classifying a novice one, is illustrated in Figure 9. Because even a same paragraph pair may be classified differently depending on the domain, the task is called domain dependent classification. Sample paragraph pairs are gathered domain by domain, and each pair is labeled with boundary or continuance. The paragraph pairs are taken from texts which are tagged with their same domain, each paragraph pair is classified into boundary or continuance. Sample paragraph pairs which are labeled with one of the two categories are encoded into string vectors in each domain.

The entire architecture of the proposed text segmentation system is illustrated in Figure 10. A text is given is the input, and paragraph pairs are generated from the text by sliding the two sized window. The sample paragraph pairs in the boundary group and the continuance group and novice paragraph pairs which are generated from the input text are encoded into string vectors, in the encoding module.

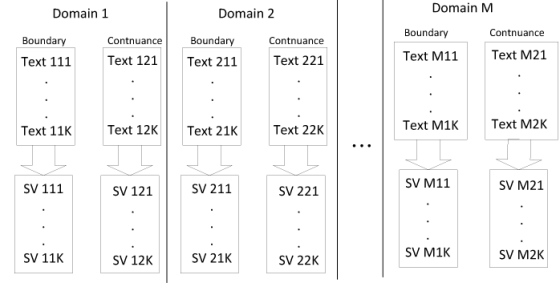


Figure 9. Preparation of Training Examples

The novice paragraph pairs are classified into one of the two categories in the similarity computation module and the voting module. The difference from the system architecture which was proposed in [12] is to replace the initial KNN algorithm by its variants.

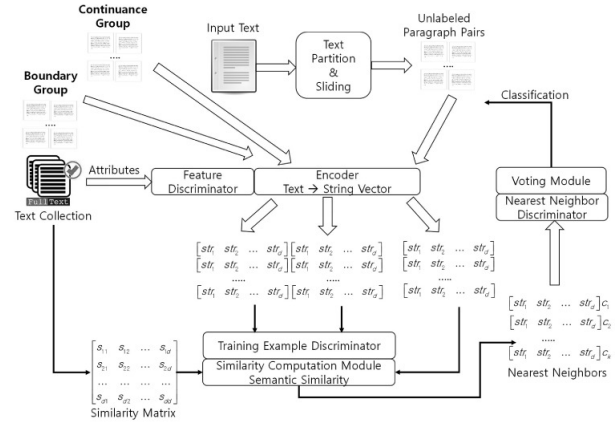


Figure 10. System Architecture

The execution process of the proposed system is illustrated as a block diagram in Figure 11. The sample paragraph pairs which are labeled with boundary or continuance are collected and encoded into string vectors. The paragraph pairs are generated from the input text by sliding the two-sized window on it, and they are encoded into string vectors. One among the three KNN variants is selected and applied to the classification of each paragraph pair into boundary or continuance. The boundary is marked between paragraphs in the pair which is classified into boundary.

Let us make some remarks on the proposed system whose architecture is illustrated in Figure 10. The M domains are predefined, in each domain, sample paragraph pairs each of which is labeled with boundary or continuance are prepared, and they are encoded into string vectors. The paragraph pairs which are generated from a text by sliding the two sized window are classified by one of the three KNN variants which are described in Section III-C. The difference from the previous version of text segmentation system is the ability to select one of the nearest neighbor discriminations, the

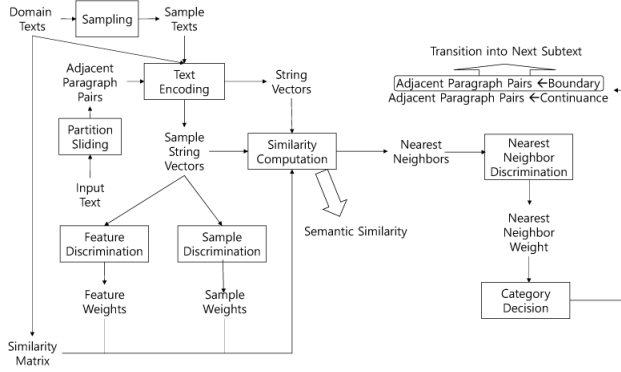


Figure 11. System Flow

attribute discriminations, and the training example discriminations. The better performance of text segmentation is expected by replacing the initial version by its variants.

IV. CONCLUSION

Let us mention the significances of this research as the conclusion. We apply the proposed similarity metric to the KNN variants as well as the standard version. We derive the three variants from the standard version by discriminating the nearest neighbors, the attributes, and the training examples. We design the text segmentation system by adopt the KNN variants with the proposed similarity metric. In the next research, we will implement it as a real system.

Let us mention the remaining tasks as the further discussions on this research. We need to define and characterize mathematically more semantic operations on string vectors. Other machine learning algorithms may be modified using the semantic operation into the string vector-based versions. The string vector based KNN variants may be applied to the connection of the text segmentation with the text categorization. The proposed KNN variants which process string vectors directly may be adopted for implementing the text segmentation system.

REFERENCES

- [1] E.H. Han, G. Karypis and V. Kumar, "Text Categorization Using Weight Adjusted k-Nearest Neighbour Classification" pp53-64, in *Advances in Knowledge Discovery and Data Mining. PAKDD 2001*, Berlin, Heidelberg:Springer, 2001.
- [2] H. Parvin, A. Hosein, and M.B. Behrouz, "MKNN: Modified k-nearest neighbor" *Proceedings of the World Congress on Engineering and Computer Science. Vol. 1*. Newswood Limited, 2008.
- [3] T. Jo, "NTC (Neural Text Categorizer): Neural Network for Text Categorization", 83-96, *International Journal of Information Studies*, 2, 2, 2010.
- [4] T. Jo, "NTSO (Neural Text Self Organizer): A New Neural Network for Text Clustering", 31-43, *Journal of Network Technology*, 1, 1, 2010.
- [5] K. Abainia, S. Ouamour, and H. Sayoud. "Neural Text Categorizer for topic identification of noisy Arabic Texts." 1-8, *The Proceedings of IEEE/ACS 12th International Conference on Computer Systems and Applications*, 2015.
- [6] T. Jo, "Long Text Segmentation by String Vector based KNN", 805-809, *The Proceedings of 18th International Conference on Advanced Communication Technology*, 2017.
- [7] T. Jo, "Improving K Nearest Neighbor into String Vector Version for Text Categorization", 1091-1097, *ICACT Transaction on Communication Technology*, 7, 1, 2018.
- [8] T. Jo, "Automatic Text Summarization using String Vector based K Nearest Neighbor", 6005-6016, *Journal of Intelligent and Fuzzy Systems*, 35, 2018.
- [9] P. P. Lakshmi and D. R. Rao, "Text classification using artificial neural networks", 603-606, *International Journal of Engineering & Technology*, 7, 1, 2018.
- [10] A.A. Nababan and O. S. Sitompul, "Attribute weighting based K-nearest neighbor using Gain Ratio", *Journal of Physics: Conference Series*, 1007, 1, 2018.
- [11] T. Jo, "Semantic String Operation for Specializing AHC Algorithm for Text Clustering", 1083-1100, *Annals of Mathematics and Artificial Intelligence*, 2019.
- [12] T. Jo, "Application of String Vector based K Nearest Neighbor to Content based Segmentation of each News Article", 58-64, *The Proceedings of 2nd International Conference on Advanced Engineering and ICT-Convergence*, 2019.
- [13] T. Jo, "Text Segmentation based on Contents using String Vector based Version of K Nearest Neighbor", in *Preparation*, 2023.